DGL-25 Kaggle Competition: *Supersoapy*, *20th*

Som Sinha, 02044826, ss2024

Nina Peire, 06010521, np324

Zimeng Zhou, 01721747, zz1224

Oliver Shakespeare, 06031528, ols24

Michael Hodgins, 06024035, mjh24

I. Methodology & Novelty (40 points)

A. Problem description & motivation (5 points)

Graph super-resolution is the process of generating a graph with N' nodes from a graph with N nodes, where $N' > N$ [1]. The goal is to develop a model that learns to infer a higher-resolution graph from a lower-resolution graph, with additional nodes (and edges), while preserving the structure and functional properties of the original graph. The super-resolution model aims to learn the inferred mapping from Low Resolution to High Resolution graphs.

The **strength** of super-resolution processes lies in transforming low-quality data into a higher-quality output which is more informative. Image super-resolution has become increasingly popular in medical imaging [2], particularly to enhance MRI scans for better readability. This allows for detection of injury and other abnormalities which were previously not visible. This report focuses on the task of applying these advancements within brain graphs. Super-resolving such graphs directly, instead of brain scans, could allow us to expand and reconstruct brain connectome in more detail. The vast majority of research on graph super-resolution involves the brain, however, there are a broad range of applications; including political voting or social network analysis, where we may gain deeper insights into the electorate, group dynamics, and how systems of people co-exist.

While the field of graph super-resolution remains exciting, many **challenges** still remain. Firstly, the clinical nature of super resolution data makes reliably sourced datasets both expensive and impractical to obtain. Secondly, graph super-resolution tasks are often computationally expensive, requiring significant time and resources. This poses a clear disadvantage and limits the scalability. Third, when using graph based methods, the model must infer node (and edge) embeddings, to learn meaningful information, in addition to upscaling from LR to HR. Ensuring the embeddings are both semantically and topologically coherent makes this task even more challenging.

B. State-of-the art methods (3 points)

See Table I.

TABLE I
Overview of GNN-based SOTA Models

Model Name	Brief Description
STP-GSR [3]	A two-stage framework which first initialises a graph in the high-resolution space. The second stage starts by using primal-dual graph formation to represent the edges as nodes. The method then uses conventional GNN techniques to learn the "node" embeddings. The final step involves performing the inverse transformation to convert the intermediate representation back to the original high-resolution space.
Adversarial GNN [1]	AGSR-Net first learns the node feature embeddings, then uses the Super Resolution Generator to generate a HR connectome from the LR connectome. Next, the super-resolved network node feature embeddings are learnt followed by an adversarial network that promotes the generated HR connectome to match the prior distribution of the target HR connectome by attempting to distinguish between the target and predicted HR connectomes.
Magnet++ for Multi-Image Super-Resolution (MISR) [4]	MISR is capable of handling heterogenous images which are images of different scale and orientation. MISR leverages the ability of every LR image to carry a different part of the HR information. Firstly, every nth LR image with size $W_n \cdot H_n$ is converted into a set of nodes $\mathcal{V}_n$ and edges are assigned between neighbouring pixels within a radius of 1. Secondly, spline-based convolutions with continuous B-spline kernels are leveraged to allow message passing across graph nodes. Thirdly, a convolution block with skip connections is applied to counteract the vanishing gradient problem. This yields a graph-based latent representation. Fourthly, a HR graph is initialised of the nodes with encoded features but no connections. Then, upsampling is achieved using a rectangular grid of $(S \cdot W) \times (S \cdot H)$ where S is the upsampling factor, of new nodes is added over this graph. One-way directed edges are added to create a bipartite graph. Features are then computed by passing node features through spline-based convolutions over the original nodes. A final CNN-based filtering step is applied to refine the output image.

C. Main figure (4 points)

See Apppendix A for our main figure.

D. Brief overview of the proposed GNN (5 points)

The proposed GNN builds upon the STP-GSR [3] GNN for graph super-resolution, by splitting the Target Edge Initialiser into smaller dimensionality steps, adding another Graph Transformed Block in the Target Edge Initialiser, as well as including residual connections in the Dual Graph Learner.

The HR graph is first initialised in two steps of incrementally larger dimensionality. The target edge initialiser then learns an initial value for the High Resolution (HR) edges, and the dual graph learner further refines these values. The primal-dual reformulation maps the graph from a node-based topology (primal) to an edge-based topology (dual) to which a GNN is applied to update the features. The proposed residual connections use a learned parameter alpha to use a weighted combination of these initial values and the refined predicted values.

E. Innovative components (10 points)

1) Final Model Improvements: See Table II.

TABLE II
INNOVATIVE COMPONENTS OF THE PROPOSED GNN FRAMEWORK

Novel Contribution	**Rationale**
Additional Layers in Target Edge Initialiser	An additional parameter `num_layers` is introduced in the `TargetEdgeInitialiser` class to allow for a variable number of Convolution and BatchNorm layers. In the implementation, `num_layers` is set to 2. This was implemented with the aim of enhancing the expressiveness of the initial edge values. Combined with the residual connections, this should yield improved edge values and therefore improved High Resolution graphs. Additionally, this may enable hierarchical feature learning.
Splitting the Target Edge Initialiser into two steps	Instead of jumping immediately from low resolution of 160 nodes to high resolution with 268 nodes, an intermediate step of 200 nodes is introduced by splitting up the Target Edge Initialiser into two stages. This gradual transition allows the model to learn the edges and refine the edge embeddings more progressively. This increased stability in super-resolution should improve generalisation and therefore generate a higher-quality high-resolution graph.
Residual Connections	Residual connections are introduced in both the TargetEdgeInitializer and DualGraphLearner. Firstly, since multiple TransformerConv layers are introduced, the model may benefit from adding residual connections from before doing the TransformerConv layers to the final output. Otherwise, multiple layers may cause over-smoothing, resulting in indistinguishable node embeddings and thus a loss of information. Secondly, the residual in DualGraphLearner is determined by the initial prediction before refinement. The weight, determined by alpha, is therefore a learned parameter which learns the importance of the initial prediction versus the refined prediction, and outputs a weighted combination of this result. Alpha is passed through a sigmoid function to ensure it is bound between 0 and 1.

2) Novel contributions which did not improve results: Attention [5] was initially implemented on top of the STP-GSR model to aggregate the embeddings using a similarity matrix. The dot-product was used to determine the similarity between the embeddings, followed by a softmax operation to normalise attention score. Having set the non-neighbouring node embeddings to negative infinity ensures a zero output from softmax. The rationale behind this approach was that more similar nodes at low resolution should remain similarly connected at high resolution. This is essential to preserve local neighbourhood structures and graph topology. Despite the sound theory, this was unsuccessful in practice, likely due to the primal-dual conversion of our graph as well as TransformerConv in STP-GSR already using attention-based aggregation.

Generative Adversarial Network [1] was built upon the STP-GSR model, by including a discriminator to work along with the generator. The discriminator aimed to distinguish the predicted HR graph from the target HR graph. This was however unsuccessful due to the fundamental challenge in GANs of finding the correct balance between the generator and discriminator. This so-called 'mode collapse' [6] means that the generator produced a limited variety of samples that is capable of 'fooling' the discriminator. As a result, the k-fold validation yielded bad results and this contribution was omitted. A possible solution to this would have been implementing a Wasserstein Generative Adversarial Network [7] instead, which uses a linear rather than sigmoid output layer. However due to time limitations, this was not implemented.

Swapping BatchNorm and Conv order (as seen in ResNetV2 [8]): An attempt was made at moving the convolution layer from before ReLU and BatchNorm to after ReLU and BatchNorm, also known as pre-activation. He et al. [8] determined that for very deep networks this may help with gradient flow, ensure clean residual paths, and can provide better regularisation. This however was also unsuccessful, likely due to the different underlying mechanisms of GNN. Applying batchnorm on activation before convolution may distort the node feature scales, creating inconsistencies and making neighbourhood preservation less reliable.

F. *Mathematical properties of the proposed GNN (13 points)*

Permutation invariance (5 points)

a) (1 point) A graph function $\boldsymbol{f_k}$ is permutation-invariant if, for each GCN layer k, we satisfy the following for any permutation matrix $\boldsymbol{P}$: $f : \mathbb{R}^{n\times n} \times \mathbb{R}^{d_k \times n} \rightarrow \mathbb{R}^{d_{k+1}}$, $f(A, H_k) = f(P^\top AP, H_k P)$. Therefore, permuting the adjacency matrix does not change the learned node embeddings.

b) (4 points) ResTP-GSR is not permutation invariant. This is because it would not make sense for our super-resolution model to be so. Additionally, the primal-dual formulation and residual connections are permutation invariant as the dual is converted back to the primal, and the residual connection does not impact the node order. Therefore the entire model is not permutation invariant.

Permutation equivariance (5 points)

a) (1 point) Permutation equivariance requires the output to be equivariant with respect to permutations of the node indices. This means that $f(t(x)) = t(f(x))$ where f is a function on our image and t is a transformation applied to the image. Node permutation equivariance can be defined mathematically as

$$H_{k+1}P = F[H_k P, P^\top AP, \phi_k]$$

b) (4 points) The proposed GNN is permutation equivariant. If a permutation is applied to one input node, this permutation should be evident in the output node. The primal-dual formulation of STP-GSR is such that if the nodes are permuted in the primal, the dual will be equally permuted. When we convert back to the primal, this permutation should remain, such that the output is equivariant with respect to the input and node permutation.

Expressiveness (3 points)

a) (1 point) An expressive GNN is capable of mapping nodes with different local neighbourhoods to different embeddings. It achieves this by using an injective neighbourhood aggregation at each layer k, ensuring that structurally different nodes are embedded differently.

b) (2 points) The proposed GNN is highly expressive because it learns both node and edge embeddings through TransformerConv. TransformerConv inherently uses attention as aggregation method, which may be injective depending on the diversity of node features. If different neighbourhoods yield similar attention weights, the softmax operation may yield indistinguishable outputs, such that it is not injective, reducing expressiveness. However, due to both node and edge embeddings being learnt, the different neighbourhoods should output different weights, such that the model is highly expressive.

II. Experimental Setup & Evaluation (27 points)

A. *Results (9 points)*

a) See Table III.

b) From the plots in Figure 1, we see that ResTP-GSR generalises well across all all folds for seven out of the eight evaluation metrics. This is inferred from the the low standard deviations in all metrics except for the small-worldness measure. The first two folds exhibit a similar small-worldness MAE, however the third fold has a relatively larger MAE. This may be due to random variance, where the input samples in the the third fold have drastically different network topologies to the other folds. To further investigate this, one may split the training data into more folds, thus reducing the effect of a single training fold's variance. Another notable feature about these plots is the relatively high Pearson Correlation Coefficient, `PCC`, values (around 0.6), which remain stable across the three folds. This suggests that the model is generalisable and does not overfit to any single training fold data. The topological measures, `MAE_CC` and `MAE_SW` remain low across all folds, suggesting that the model accurately preserves graph topology.

c) Empirically, training the model for 30 epochs was found to offer the best balance between convergence and overfitting. This ensured that the model learned meaningful patterns without excessively memorising the training data. To monitor the RAM usage of the model during training, a simple script ran alongside the training script. Given the specific OS-provided process ID of the training process, the monitoring script used the package `psutil` to monitor

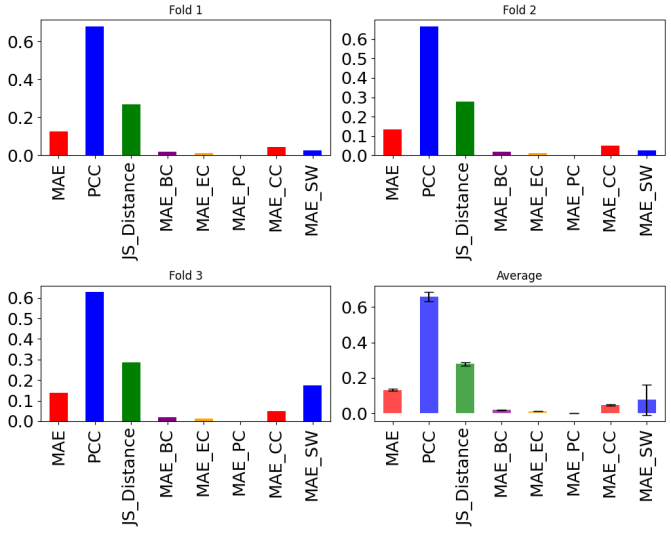

Fig. 1. Bar-plots of eight evaluation metrics of ResTP-GSR across three folds. A bar-plot showing the aggregated performance across the three folds with the standard-deviations is also provided.

the memory usage over the life-time of the training process. The plot is shown in Figure 2. We see that the memory usage stays level throughout the duration of the epoch signalling that there is no memory leak occurring during training. The three spikes shown correspond to the start of a fold, this spike can be attributed to initially loading the model at the start of the epoch, in particular, the dual graph construction. The total runtime of the training process is 810 seconds. This is roughly 7 times slower compared to the MLP as seen in Figure 6

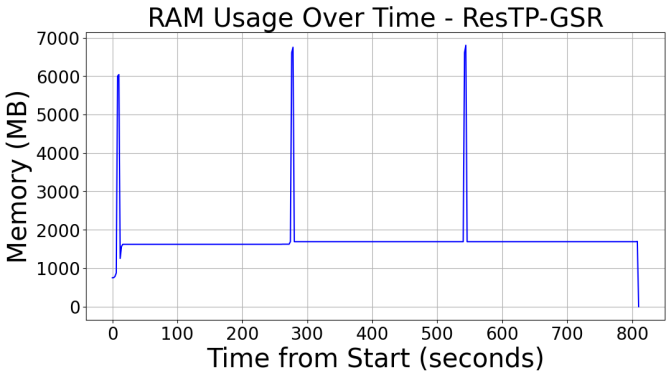

Fig. 2. A plot of memory usage across the training process lifetime.

d) When retraining a model for submission, there is no to test on each epoch iteration, as our defined model should have the most appropriate parameters from pre-tuning. Given this, the train_LR does not need to be split into a train and validation set, and the model can train while exposed to the full train data in the low and high resolution space. The test_lr can then be passed through the model to generate predictions. To follow the competition guidelines, each matrix is flattened, and each vector represented graph is flattened again into one large vector. An index column is further added to the dataframe, leaving it readdy for submission. Final Kaggle score on private Kaggle test: 0.158, Position: 20th.

TABLE III
ADDITIONAL TOPOLOGICAL MEASURES

Measure Name	**Brief Description & Rationale**
Clustering Coefficient	The clustering coefficient measures how well nodes in a graph tend to cluster together. From [9], for some node u, this is expressed mathematically as: $c_u = \frac{2T(u)}{deg(u)(deg(u)-1)}$ where $T(u)$ is defined as the number of triangles which node u is a part of and $deg(u)$ is the degree of u. By using this measure to evaluate the super- resolved high- resolution graph, we have a measure of how well the model replicates the desired local clustering properties of the ground truth high-resolution graphs
Small-worldness	Brains are known to exhibit small-world properties [10], where there are several, strongly-connected modular networks. These sub-networks are still able to communicate through inter-module edges. This architecture balances information segregation and integration. The small-worldness of graphs is calculated using the following procedure: 1) Compute the average shortest path length L of the given graph. 2) Compute the clustering coefficient C of the given graph. 3) Generate a reference small-world graph using the Watts-Strogatz model. 4) Compute the average shortest path length L_{rand} of the reference graph. 5) Compute the clustering coefficient C_{rand} of the reference graph. 6) Calculate the small-worldness coefficient SW using the formula: $SW = \frac{\frac{L}{C}}{\frac{L_{\text{rand}}}{C_{\text{rand}}}}$

B. *Comparison Against Other Methods (6 points)*

From Figure 3, we see that ResTP-GSR outperforms both the state-of-the-art model (STP-GSR) and the naive MLP across key metrics such as `MAE`, `PCC` and `JS_Distance`, suggesting that a) ResTP-GSR makes more accurate predictions b) ResTP-GSR produces HR graphs with the strongest correlation to the ground truth HR graphs and c) ResTP-GSR preserves the graph distribution more strongly than the other two models. Furthermore, ResTP-GSR's superior performance in measures such as `MAE_BC` and `MAE_EC` suggests that ResTP-GSR preserves node importance better than the other models. The PageRank centrality measures the importance of a node as a function of how likely a random walker would visit the node [11]. Thus, ResTP-GSR performing better than the other two models

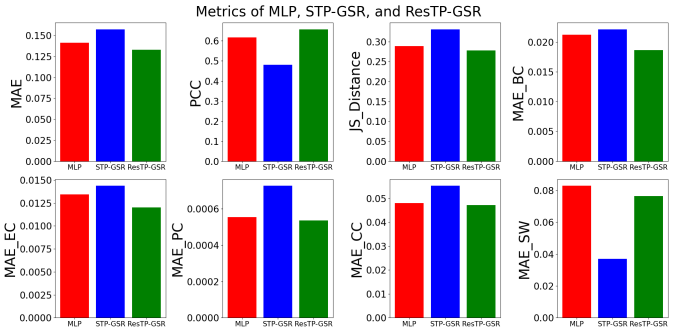

Fig. 3. Plots comparing the performance of the naive MLP model, STP-GSR and ResTP-GSR across the eight evaluation metrics, averaged across all folds. See appendix figure 5 for enlarged version.

in this metric, along with the aforementioned superior performance in `MAE_EC` and `MAE_BC` signals that ResTP-GSR is more effective in retaining the global importance of nodes.

Given that ResTP-GSR implements enhancements on top of the already-existing STP-GSR model, its improved performance in most metrics displayed in Figure 3 reflect our expectations as it was designed as an enhancements to the already-existing STP-GSR model. However, the MLP and STP-GSR models did exhibit certain surprising results. For example, the STP-GSR model had a higher `JS_Distance` compared to the MLP model. Ordinarily, one would expect the STP-GSR to have a much lower `JS_Distance` than the MLP as the STP-GSR should be able to explicitly model the graph structure. One possible explanation is that the STP-GSR may slightly distort the edge weights, thus affecting the distribution similarity.

C. Scalability of Your Proposed GNN Model (7 points)

The model makes use of multiple `TransformerConv` layers in both phases of training. These layers scale quadratically with respect to the number of nodes, and linearly with respect to the number of edges. This is because transformer's rely on an attention mechanism which scales quadratically [12]. As we are super-resolving the graph and therefore including more nodes, the model will scale poorly with increased graph complexity. Given a larger dataset, we hypothesise that the training time and memory will scale linearly with the number of input graphs.

D. Reproducibility of Your Proposed GNN Model (5 points)

Reproducibility is not guaranteed due to the inherent randomness of PyTorch [13]. In particular, the random weights and biases initialisation means that training is not reproducible. Once training is completed, the model is saved in a .pt file which is then used for testing. This means that test outputs should be reproducible. As a result, upon repeat, training runs may vary in output. However, the model yields sufficiently stable results that this does not pose a problem. This consistency can be observed in Figure 1, where each Fold yields the same metrics except for Small World-ness, within a 10^{-3} error margin. Reproducibility can be forced by setting a manual seed in torch, however, as the documentation [13] states, even a manual seed cannot guarantee complete reproducibility in PyTorch.

III. Discussion & Reflections (8 points)

a) See Table IV

TABLE IV
Strengths & Weaknesses of the Proposed GNN

Strength 1	While the baseline STP-GSR performs strongly on the topological measures, it underperforms on the MAE metric [1]. A major strength and improvement of our ResTP-GSR model is the use of deeper GNN layers coupled with residual connections help the model acheive a lower MAE score. This can be seen in Figure 3, where our model achieves an average MAE score over 3-folds of 0.1328, compared to 0.1572 of STP-GSR. In fact, our model performs better on all metrics except the small-worldness MAE.
Strength 2	The dual graph representation in our model focuses explicitly on edge relationships. This is beneficial for brain graphs where connectivity patterns (edges) are often more informative than individual node properties. This allows our model to better predict brain graph edges.
Weakness 1	Our model is computationally more expensive to train and perform inference on. This can be seen by the increased time required to train, as well as the increased RAM utilisation. In comparison, the baseline STP-GSR model uses less memory than ResTP-GSR, this can be seen in Figure 7. In the context of brain graph super-resolution, scalability may become an issue for our mode. If brain graphs become larger/are more detailed, along with datasets growing larger, our model may become computationally infeasible to use.
Weakness 2	Our model does not explicitly incorporate biological constraints or prior knowledge about brain connectivity. For example when we initialize edge features for our of the HR target graph in the TargetEdgeInitializer, we perform a simple matrix multiplication, which could be further enhanced to generate connections that are anatomically implausible in brain networks.

b) (4 points)

Improvements:

- We notice the model is prone to overfitting. Early stopping could improve this by monitoring how well the model does against the validation data and stop once the model no longer improves on the validation set.
- We could also address the problem of computational inefficiency by exploring quantization during training/inference and evaluate the trade-off between computational savings and model accuracy.
- Our model does not explicitly incorporate the biological constraints. To address this, we could incorporate these constraints into the loss function. Currently we are calculating the L1-loss against brain graph edges. However, if we add a term to the loss that penalises edges that violate anatomical constraints, we could output brain graphs that take into account anatomy.

References

[1] M. Isallari and I. Rekik, ''Brain graph super-resolution using adversarial graph neural network with application to functional brain connectivity,'' *Medical Image Analysis*, vol. 71, p. 102084, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361841521001304

[2] E. Van Reeth, I. W. K. Tham, C. H. Tan, and C. L. Poh, ''Super-resolution in magnetic resonance imaging: A review,'' *Concepts in Magnetic Resonance Part A*, vol. 40A, no. 6, pp. 306–325, 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cmr.a.21249

[3] P. Singh and I. Rekik, *Strongly Topology-Preserving GNNs for Brain Graph Super-Resolution*. Springer Nature Switzerland, Oct. 2024, p. 124–136. [Online]. Available: http://dx.doi.org/10.1007/978-3-031-74561-4_11

[4] T. Tarasiewicz and M. Kawulok, ''A graph neural network for heterogeneous multi-image super-resolution,'' *Pattern Recognition Letters*, vol. 189, pp. 214–220, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865525000297

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, ''Attention is all you need,'' 2023. [Online]. Available: https://arxiv.org/abs/1706.03762

[6] R. Zhang, ''Generative adversarial networks series 2 — the balancing act: Training techniques and challenges,'' February 2024, accessed: 2025-03-10. [Online]. Available: https://rendazhang.medium.com/generative-adversarial-networks-series-2-the-balancing-act-training-techniques-and-challenges-a6d24f6118dc

[7] M. Arjovsky, S. Chintala, and L. Bottou, ''Wasserstein gan,'' 2017. [Online]. Available: https://arxiv.org/abs/1701.07875

[8] K. He, X. Zhang, S. Ren, and J. Sun, ''Identity mappings in deep residual networks,'' 2016. [Online]. Available: https://arxiv.org/abs/1603.05027

[9] NetworkX Developers, ''NetworkX clustering documentation,'' 2024, accessed: March 11, 2025. [Online]. Available: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.cluster.clustering.html

[10] X. Liao, A. V. Vasilakos, and Y. He, ''Small-world human brain networks: Perspectives and challenges,'' *Neuroscience & Biobehavioral Reviews*, vol. 77, pp. 286–300, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0149763416307849

[11] A. Disney, ''Eigencentrality vs pagerank: What's the difference?'' jan 2020, accessed: 2025-03-11. [Online]. Available: https://cambridge-intelligence.com/eigencentrality-pagerank/

[12] F. D. Keles, P. M. Wijewardena, and C. Hegde, ''On the computational complexity of self-attention,'' 2022. [Online]. Available: https://arxiv.org/abs/2209.04881

[13] PyTorch Documentation Team, ''Reproducibility,'' 2024, accessed: 2025-03-11. [Online]. Available: https://pytorch.org/docs/stable/notes/randomness.html

Appendix

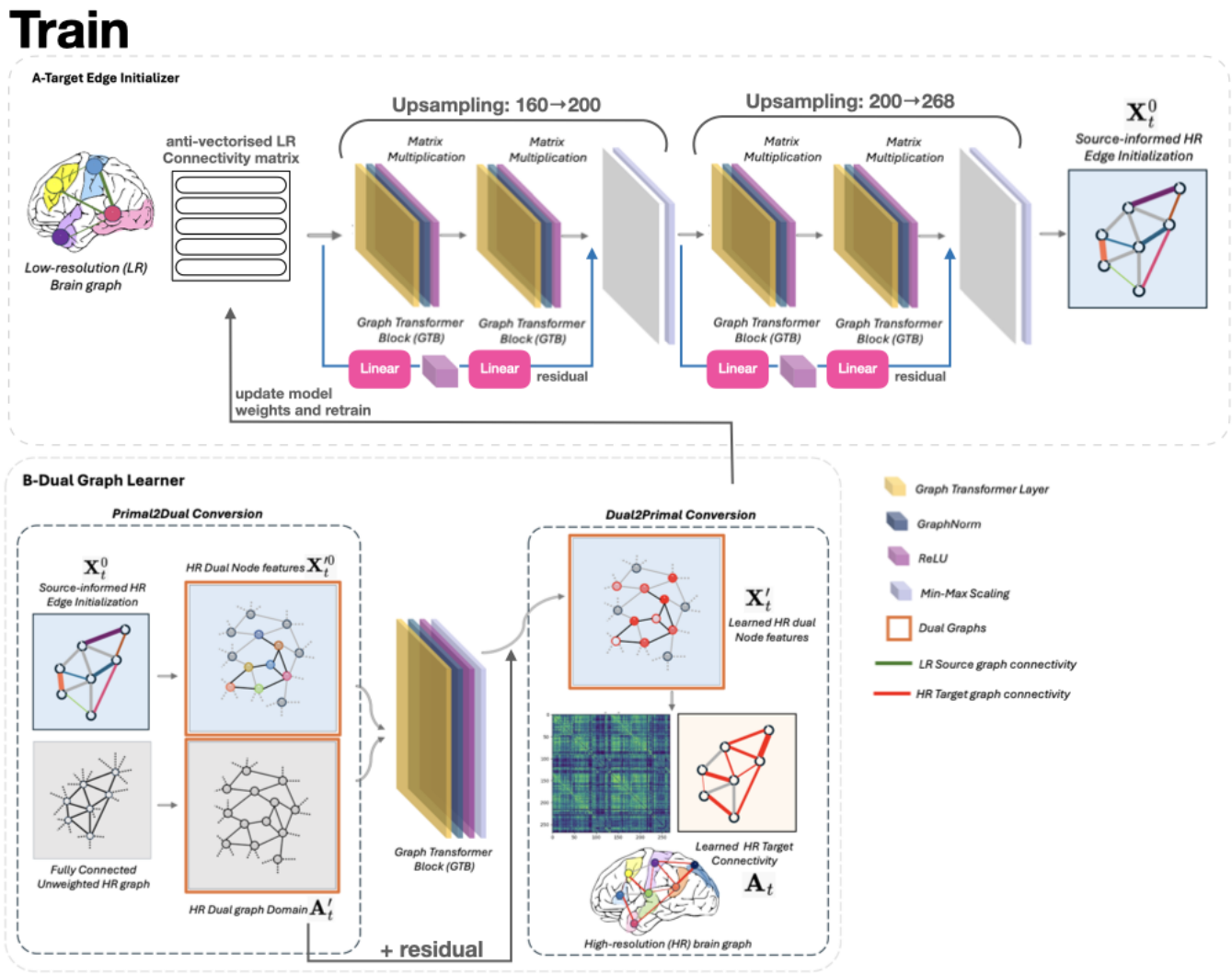

Fig. 4. Essential steps of the proposed solution (adapted from [3])

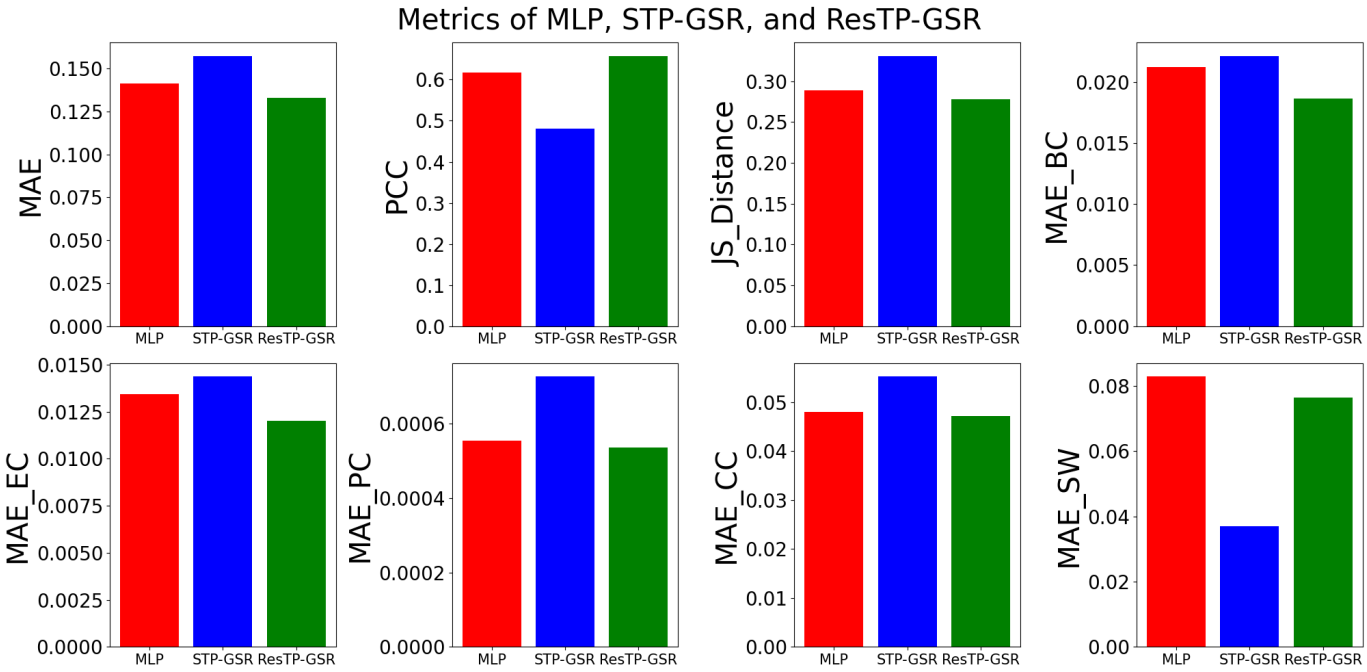

Fig. 5. Plots comparing the performance of the naive MLP model, STP-GSR and ResTP-GSR across the eight evaluation metrics, averaged across all folds.

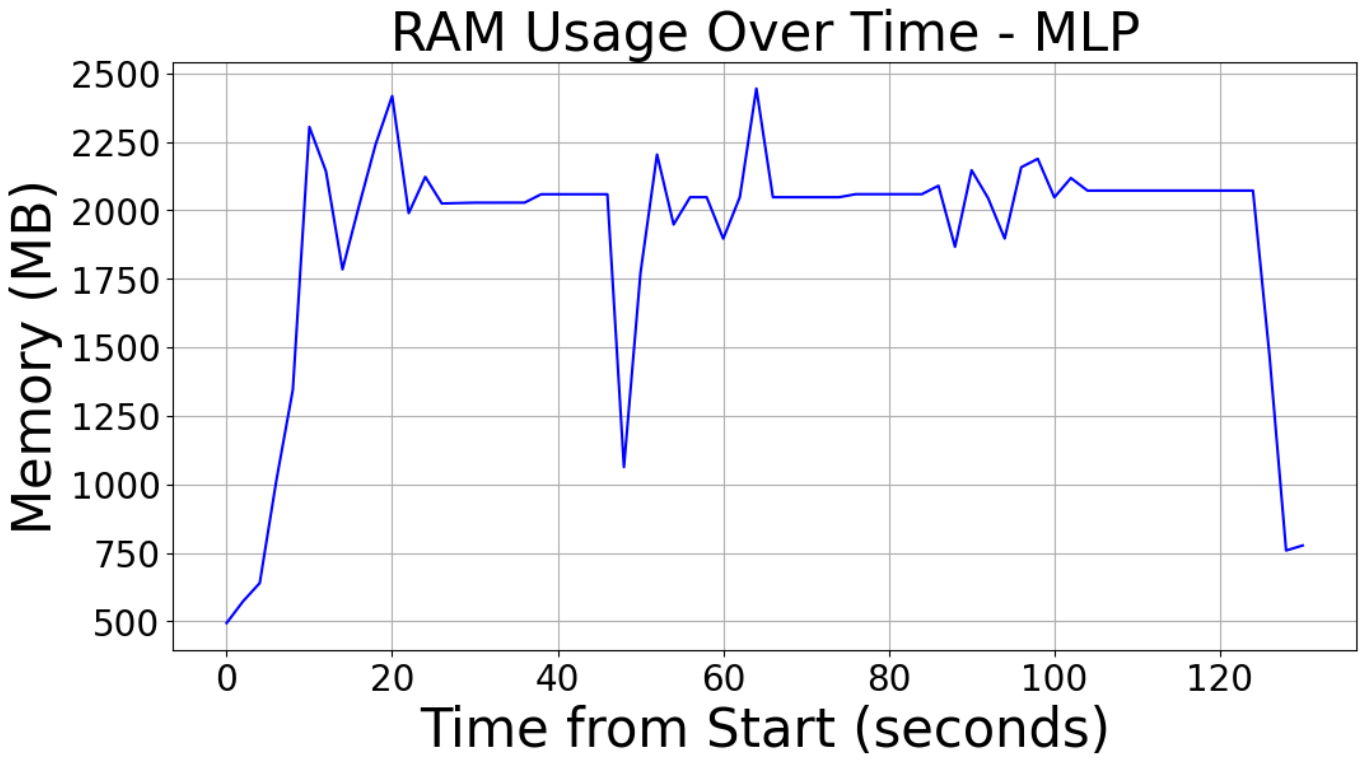

Fig. 6. RAM usage for the MLP

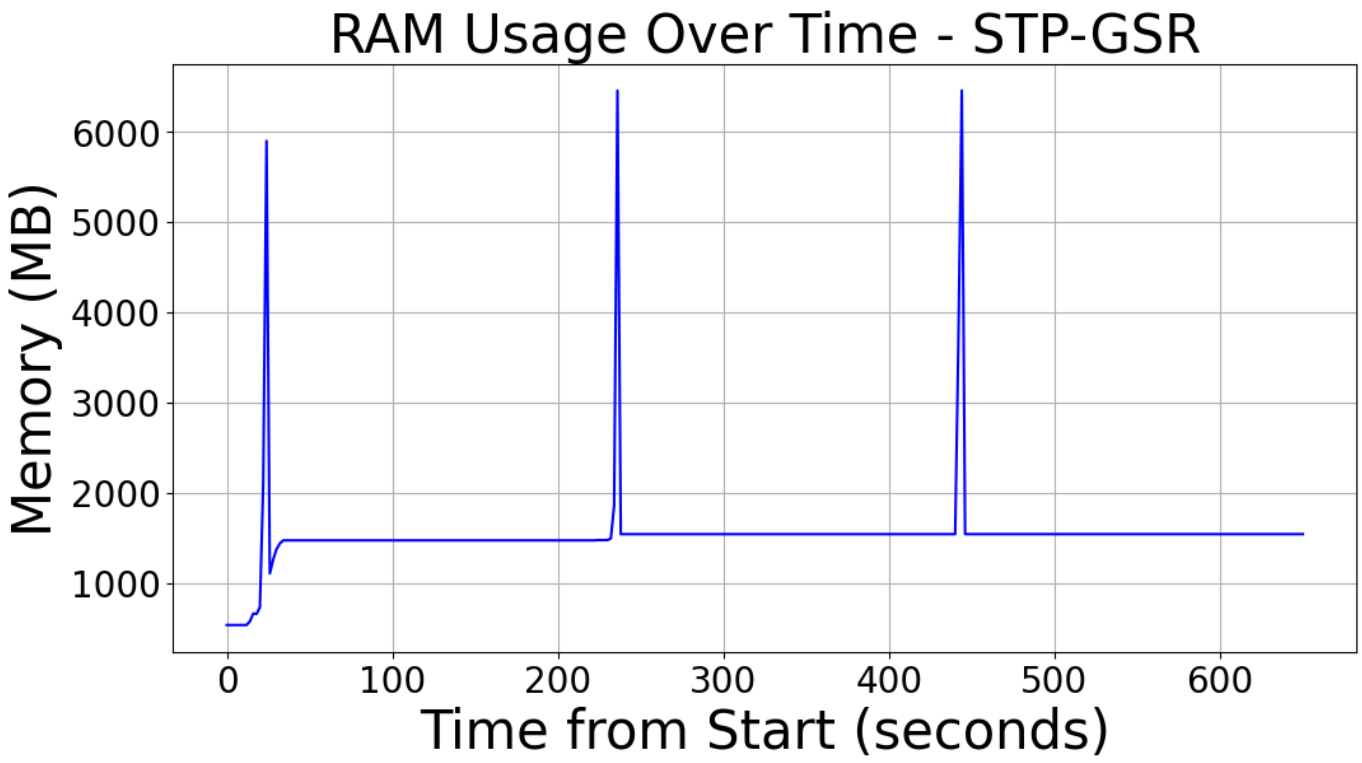

Fig. 7. RAM usage for the baseline STP GSR model.